Assignment -2

1.explain three dimensional data index?

Ans) Python has given us every solution that we might require. Python has many methods predefined in it. These methods help us to add an element to a given list. Python does not support the array fully. At this point, to get simpler with the array, we need to make use of function insert.

Example:

```
mylist = [[['@', '@'], ['@', '@']], [['@', '@'], ['@', '@']], [['@', '@'], ['@', '@']]]

# number tuple

addition = ['$','$']

# inserting $ symbol in the existing list

my list.insert(2, addition)

print('Updated List is: ', mylist)
```

Output:

```
Updated List is:  [[['@', '@'], ['@', '@']], [['@', '@'], ['@', '@']], ['$', '$'], [['@', '@'], ['@', '@']]]
```

Here, in the above program, we are inserting a new array element with the insert method's help, which python provides. In the above program, we have one 3 dimensional lists called my list.

The insert method takes two arguments. One is position, i.e. nothing but the index number. And second is an actual element you want to insert in the existing array or a list. Here, we took the element in one variable which we wanted to insert. We are applying the insert method on mylist.

Try to execute this program. Play with the output for different combinations. In the above program, we have given the position as 2. We all know that the array index starts at zero (0). That means a new element got added into the 3rd place, as you can see in the output.

2.whats the difference between a series and a data frame?

Ans) series:Series is a type of list which can take integer values, string values, double value and more. Series can only contain single list with index, whereas dataframes can be made of more than one series or we can say that a dataframes is a collection of series that can be used to analyse the data.

Series is a one dimensional object that can any datatype , it can be int , float , string or anything. In simpler words series are somewhat like a list , but here you can define your index too .

Instead of starting from 0 to len(n-1) , you can define your own index like a, b ,c which are similar to keys in dictionary.

Now DataFrame is a two dimensional object. It has columns and rows. Each column may have different types of values including list , tuples , dictionaries and even other dataframes too.

data frame:A pandas dataframe is a two-dimensional data-structure that can be thought of as a spreadsheet. A dataframe can also be thought of as a combination of two or more series.

3.what role does pandas play in data cleaning?

Ans) data cleaning:

When working with multiple data sources, there are many chances for data to be incorrect, duplicated, or mislabeled. If data is wrong, outcomes and algorithms are unreliable, even though they may look correct. Data cleaning is the process of changing or eliminating garbage, incorrect, duplicate, corrupted, or incomplete data in a dataset. There's no such absolute way to describe the precise steps in the data cleaning process because the processes may vary from dataset to dataset. Data cleansing, data cleansing, or data scrub is that the initiative among the general data preparation process. Data cleaning plays an important part in developing reliable answers and within the analytical process and is observed to be a basic feature of the info science basics. The motive of data cleaning services is to construct uniform and standardized data sets that enable data analytical tools and business intelligence easy access and perceive accurate data for each problem.

data cleaning essintial for:

Data cleaning is the most important task that should be done as a data science professional. Having wrong or bad quality data can be detrimental to processes and analysis. Having clean data will ultimately increase overall productivity and permit the very best quality information in your decision-making.

1. Error-Free Data: When multiple sources of data are combined there may be chances of so much error. Through Data Cleaning, errors can be removed from data. Having clean data which is free from wrong and garbage values can help in performing analysis faster as well as efficiently. By doing this task our considerable amount of time is saved. If we use data containing garbage values, the results won't be accurate. When we don't use accurate data, surely we will make mistakes. Monitoring errors and good reporting helps to find where errors are coming from, and also makes it easier to fix incorrect or corrupt data for future applications.

2. Data Quality: The quality of the data is the degree to which it follows the rules of particular requirements. For example, if we have imported phone numbers data of different customers, and in some places, we have added email addresses of customers in the data. But because our needs were straightforward for phone numbers, then the email addresses would be invalid data. Here some pieces of data follow a specific format. Some types of numbers have to be in a specific range. Some data cells might require a selected quite data like numeric, Boolean, etc. In

every scenario, there are some mandatory constraints our data should follow. Certain conditions affect multiple fields of data in a particular form. Particular types of data have unique restrictions. If the data isn't in the required format, it would always be invalid. Data cleaning will help us simplify this process and avoid useless data values.

3. Accurate and Efficient: Ensuring the data is close to the correct values. We know that most of the data in a dataset are valid, and we should focus on establishing its accuracy. Even if the data is authentic and correct, it doesn't mean the data is accurate. Determining accuracy helps to figure out the data entered is accurate or not. For example, the address of a customer is stored in the specified format, maybe it doesn't need to be in the right one. The email has an additional character or value that makes it incorrect or invalid. Another example is the phone number of a customer. This means that we have to rely on data sources, to cross-check the data to figure out if it's accurate or not. Depending on the kind of data we are using, we might be able to find various resources that could help us in this regard for cleaning.

4. Complete Data: Completeness is the degree to which we should know all the required values. Completeness is a little more challenging to achieve than accuracy or quality. Because it's nearly impossible to have all the info we need. Only known facts can be entered. We can try to complete data by redoing the data gathering activities like approaching the clients again, re-interviewing people, etc. For example, we might need to enter every customer's contact information. But a number of them might not have email addresses. In this case, we have to leave those columns empty. If we have a system that requires us to fill all columns, we can try to enter missing or unknown there. But entering such values does not mean that the data is complete. It would be still being referred to as incomplete.

5. Maintains Data Consistency: To ensure the data is consistent within the same dataset or across multiple datasets, we can measure consistency by comparing two similar systems. We can also check the data values within the same dataset to see if they are consistent or not. Consistency can be relational. For example, a customer's age might be 25, which is a valid value and also accurate, but it is also stated as a senior citizen in the same system. In such cases, we have to cross-check the data, similar to measuring accuracy, and see which value is true. Is the client a 25-year old? Or the client is a senior citizen? Only one of these values can be true. There are multiple ways to for your data consistent.

4. how do you use pandas to make data frames out of n dimensional array?

Ans) I want to be able to create n-dimensional dataframes. I've heard of a method for 3D dataframes using panels in pandas but, if possible, I would like to extend the dimensions past 3 dims by combining different datasets into a super dataframe

I tried this but I cannot figure out how to use these methods with my test dataset -> Constructing 3D Pandas DataFrame

Also, this did not help for my case -> Pandas Dataframe or Panel to 3d numpy array

I made a random test dataset with arbitrary axis data trying to mimic a real situation; there are 3 axis (i.e. patients, years, and samples). I tried adding a bunch of dataframes to a list and then making a dataframe with that but it didn't work :(I even tried a panel as in the 2nd link above but I couldn't get it to work either.

program:

```
#Reproducibility

np.random.seed(1618033)


#Set 3 axis labels/dims

axis_1 = np.arange(2000,2010) #Years

axis_2 = np.arange(0,20) #Samples

axis_3 = np.array(["patient_%d" % i for i in range(0,3)]) #Patients


#Create random 3D array to simulate data from dims above

A_3D = np.random.random((years.size, samples.size, len(patients))) #(10, 20, 3)


#Create empty list to store 2D dataframes (axis_2=rows, axis_3=columns) along axis_1

list_of_dataframes=[]


#Iterate through all of the year indices

for i in range(axis_1.size):

   #Create dataframe of (samples, patients)

   DF_slice = pd.DataFrame(A_3D[i,:,:],index=axis_2,columns=axis_3)

   list_of_dataframes.append(DF_slice)

#    print(DF_slice) #preview of the 2D dataframes "slice" of the 3D array

#        patient_0  patient_1  patient_2

#     0   0.727753   0.154701   0.205916

#     1   0.796355   0.597207   0.897153

#     2   0.603955   0.469707   0.580368

#     3   0.365432   0.852758   0.293725

#     4   0.906906   0.355509   0.994513

#     5   0.576911   0.336848   0.265967

#    ...
#     19   0.583495   0.400417   0.020099

# DF_3D = pd.DataFrame(list_of_dataframes,index=axis_2, columns=axis_1)
# Error
# Shape of passed values is (1, 10), indices imply (10, 20)
```

5. explain the notations of pandas plotting?

Ans) arameters

dataSeries or DataFrame

The object for which the method is called.

xlabel or position, default None

Only used if data is a DataFrame.

ylabel, position or list of label, positions, default None

Allows plotting of one column versus another. Only used if data is a DataFrame.

kindstr

The kind of plot to produce:

'line' : line plot (default)

'bar' : vertical bar plot

'barh' : horizontal bar plot

'hist' : histogram

'box' : boxplot

'kde' : Kernel Density Estimation plot

'density' : same as 'kde'

‘area’ : area plot

‘pie’ : pie plot

‘scatter’ : scatter plot (DataFrame only)

‘hexbin’ : hexbin plot (DataFrame only)

axmatplotlib axes object, default None

An axes of the current figure.

subplotsbool, default False

Make separate subplots for each column.

sharexbool, default True if ax is None else False

In case subplots=True, share x axis and set some x axis labels to invisible; defaults to True if ax is None otherwise False if an ax is passed in; Be aware, that passing in both an ax and sharex=True will alter all x axis labels for all axis in a figure.

shareybool, default False

In case subplots=True, share y axis and set some y axis labels to invisible.

layouttuple, optional

(rows, columns) for the layout of subplots.

figsizea tuple (width, height) in inches

Size of a figure object.

use_indexbool, default True

Use index as ticks for x axis.

titlestr or list

Title to use for the plot. If a string is passed, print the string at the top of the figure. If a list is passed and subplots is True, print each item in the list above the corresponding subplot.

gridbool, default None (matlab style default)

Axis grid lines.

legendbool or {'reverse'}

Place legend on axis subplots.

stylelist or dict

The matplotlib line style per column.

logxbool or 'sym', default False

Use log scaling or symlog scaling on x axis. .. versionchanged:: 0.25.0

logybool or 'sym' default False

Use log scaling or symlog scaling on y axis. .. versionchanged:: 0.25.0

loglogbool or 'sym', default False

Use log scaling or symlog scaling on both x and y axes. .. versionchanged:: 0.25.0

xtickssequence

Values to use for the xticks.

ytickssequence

Values to use for the yticks.

xlim2-tuple/list

Set the x limits of the current axes.

ylim2-tuple/list

Set the y limits of the current axes.

xlabellabel, optional

Name to use for the xlabel on x-axis. Default uses index name as xlabel, or the x-column name for planar plots.

New in version 1.1.0.

Changed in version 1.2.0: Now applicable to planar plots (scatter, hexbin).

ylabellabel, optional

Name to use for the ylabel on y-axis. Default will show no ylabel, or the y-column name for planar plots.

New in version 1.1.0.

Changed in version 1.2.0: Now applicable to planar plots (scatter, hexbin).

rotint, default None

Rotation for ticks (xticks for vertical, yticks for horizontal plots).

fontsizeint, default None

Font size for xticks and yticks.

colormapstr or matplotlib colormap object, default None

Colormap to select colors from. If string, load colormap with that name from matplotlib.

colorbarbool, optional

If True, plot colorbar (only relevant for 'scatter' and 'hexbin' plots).

positionfloat

Specify relative alignments for bar plot layout. From 0 (left/bottom-end) to 1 (right/top-end). Default is 0.5 (center).

tablebool, Series or DataFrame, default False

If True, draw a table using the data in the DataFrame and the data will be transposed to meet matplotlib's default layout. If a Series or DataFrame is passed, use passed data to draw a table.

yerrDataFrame, Series, array-like, dict and str

See Plotting with Error Bars for detail.

xerrDataFrame, Series, array-like, dict and str

Equivalent to yerr.

stackedbool, default False in line and bar plots, and True in area plot

If True, create stacked plot.

sort_columnsbool, default False

Sort column names to determine plot ordering.

secondary_ybool or sequence, default False

Whether to plot on the secondary y-axis if a list/tuple, which columns to plot on secondary y-axis.

mark_rightbool, default True

When using a secondary_y axis, automatically mark the column labels with "(right)" in the legend.

include_boolbool, default is False

If True, boolean values can be plotted.

backendstr, default None

Backend to use instead of the backend specified in the option plotting.backend. For instance, 'matplotlib'. Alternatively, to specify the plotting.backend for the whole session, set pd.options.plotting.backend.

Returns

matplotlib.axes.Axes or numpy.ndarray of them

If the backend is not the default matplotlib one, the return value will be the object returned by the backend.